# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-021031
(43)Date of publication of application : 24.01.1995

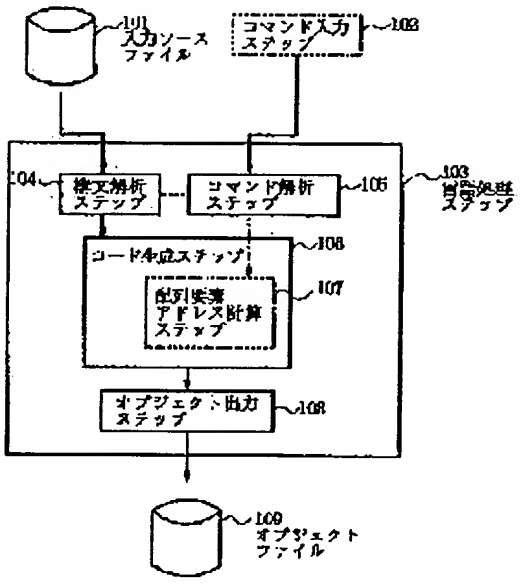
51)Int.Cl. G06F 9/45

21)Application number : 05-163384
22)Date of filing : 01.07.1993

(71)Applicant : NEC CORP
(72)Inventor : ISOZAKI HIROKO

## 54) LANGUAGE PROCESSING METHOD

57)Abstract:
'URPOSE: To accelerate the execution of an object to be generated, and to shorten he size by analyzing command designation and selecting any optimum calculating nethod concerning the address calculating method of arranged elements.
:ONSTITUTION: In a command analysis step 105, the command designation to esignate the size of arrangement is analyzed, when the reference of arranged elements ; recognized in a syntax analysis step 104, on the other hand, in an arranged element ddress calculation step 107, calculation for calculating offset from the head of rrangement is performed by selecting any suitable data width based on command nformation analyzed in advance, and an efficient arranged element address calculation bject is generated in an object output step.




EGAL STATUS

Date of request for examination] 01.07.1993
Date of sending the examiner's decision of rejection] 05.01.1999
Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
Date of final disposal for application]
Patent number] 3233245
Date of registration] 21.09.2001
Number of appeal against examiner's decision of rejection] 11-01732
Date of requesting appeal against examiner's decision of rejection] 04.02.1999
Date of extinction of right] 21.09.2004

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－21031

(43)公開日　平成７年(1995)１月24日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | 技術表示箇所 |
|---|---|---|---|---|
| Ｇ０６Ｆ　9/45 | | | | |
| | | 9292－5B | Ｇ０６Ｆ　9/44　322　Ｈ | |

審査請求　有　　請求項の数１　ＯＬ　（全　８　頁）

(21)出願番号　特願平5－163384

(22)出願日　平成５年(1993)７月１日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　磯崎　博子
東京都港区芝五丁目７番１号　日本電気株式会社内

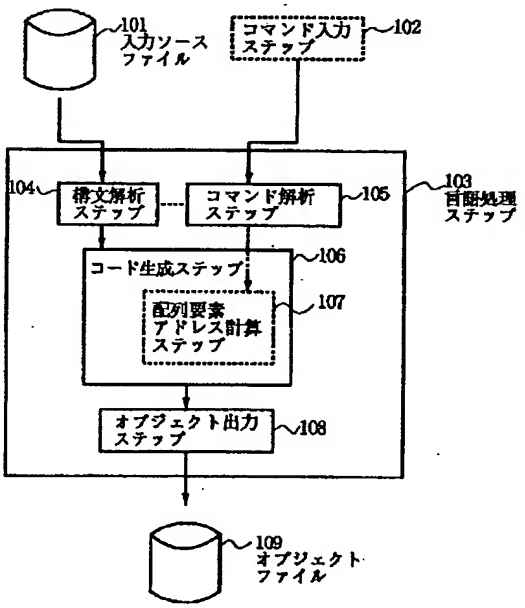
(74)代理人　弁理士　京本　直樹　（外２名）

(54)【発明の名称】　言語処理方法

(57)【要約】

【目的】配列要素のアドレス計算方法において、コマンド指定を解析し最適な計算方法を選択することにより、生成するオブジェクトの実行時間の向上とサイズの短縮を図る。

【構成】コマンド解析ステップ１０５で配列の大きさを指定したコマンド指定を解析する一方、配列要素の参照を構文解析ステップ１０４で認識したならば、配列要素アドレス計算ステップ１０７において先に解析したコマンド情報に基づき配列の先頭からのオフセットを求める計算を適切なデータ幅を選択して行ない、効率の良い配列要素アドレス計算オブジェクトをオブジェクト出力ステップで生成する。



1

【特許請求の範囲】

【請求項１】　ソースプログラム情報を入力するステップと、前記ソースプログラム情報を解析するステップと、オブジェクト情報の生成時に処理する配列のアドレス計算をするステップと、前記オブジェクト情報を出力するステップとを有する言語処理プログラムにおいて、前記配列のアドレス計算をするステップが配列の最大サイズを指定したコマンド入力を解析して得たコマンド情報から最短のデータ幅を選択するステップと、前記選択したデータ幅で配列の先頭からのオフセットを求める計算を行なうステップとを有することを特徴とする言語処理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、言語処理方法に関し、特に配列構造を有する言語の言語処理方法に関する。

【０００２】

【従来の技術】言語処理プログラムが入力したソースプログラムを翻訳し、その結果として生成したファイルをオブジェクトモジュール（以下、オブジェクトと呼ぶ）と言う。一般に、言語処理プログラムによって処理されるソフトウェアはその実行性能として速さが求められる。これを実現するためには言語処理を行なった結果であるオブジェクトのサイズが小さいこと、およびオブジェクトの実行が早いことが必要である。一方、ソフトウェアの規模の増大に伴い、ハードウェアのデータ空間は拡大する方向にある。

【０００３】従来の言語処理方法では、このデータ空間の拡大に伴い、配列要素のデータ空間中の位置を計算するアドレス計算に関しても、データ空間の大きさに合せて行なっている。仮に、データ空間を４ギガバイト（＝$2^{32}$バイト）だとすると、このデータ空間のアドレスを表現するには３２ビット＝４バイト必要である。この仮定に基づき、図９を参照して従来の配列要素のアドレス計算の方法について説明する。

【０００４】配列要素の先頭からのオフセットを計算する処理では、まず配列の添字を取り出す（ステップ９０１）。次に添字の内容を４バイトに拡張する（ステップ９０２）。そして、拡張後の添字の内容（４バイト）と配列要素のデータ幅を乗算してオフセットを求める（ステップ９０３）。以上がオフセットの計算の処理である。

【０００５】次に、配列要素のアドレスは、配列の先頭アドレスを取り出し（ステップ９０４）、取り出した配列先頭アドレスと先に求めたオフセットを加算する（ステップ９０５）ことにより求める。

【０００６】言語プログラムは、以上の計算方法に対応したオブジェクトを生成していた。ステップ９０２で添字の内容を４バイトに拡張しているのは次の理由による。一般に高級言語では複数モジュールのリンクを可能

2

とするために、変数の外部参照を許しているが、配列も同様に外部参照した場合、配列の大きさが不明となる。そのため、求めるオフセットの最大値がデータ空間と同じ大きさを持つ可能性を考慮して４バイトに拡張する処理を行なっている。

【０００７】

【発明が解決しようとする課題】上述した従来の方式では、オフセットを求めるための乗算の前に添字を４バイトに拡張してしまうために、乗算の一方の項が４バイトとなり乗算を行なう負荷が大きくなる問題が発生する。これは、４バイトデータをオペランドとして指定可能な乗算命令を持つアーキテクチャをターゲットとしている言語処理では問題はないが、１６ビット＝２バイト、あるいはそれ以下のビット幅を中心とした命令を備えるアーキテクチャをターゲットとする場合には、２バイトデータを取り扱うのと４バイトデータを取り扱う方のとでは後者の方が命令サイズが長い。つまり、データ空間のアドレス幅よりもアーキテクチャのデータ幅が小さい場合に、前述の前列要素のアドレス計算方法をとっていると生成するオブジェクトのサイズが大きく、かつオブジェクトの実行性能が悪くなってしまう。

【０００８】このような問題を解決する方法として、１６ビットアーキテクチャマシンで一時的に取り扱うデータ空間が、１６ビットでアドレスを表現できる大きさであった。このために配列の計算においても展示を１６ビット＝２バイトに拡張すれば良く、２バイトデータともう１つのデータとの乗算でオフセットを求めることになるので、もう１つのデータが２バイトよりも小さければ一命令で乗算を行なうことが可能であった。

【０００９】しかしながら、最近ではアーキテクチャのデータ幅以上に大きなデータ空間が求められる傾向にあり、配列要素のアドレス計算においても、前述の通りオブジェクトサイズが大きく実行性能も悪いオブジェクトを生成してしまう問題が生じている。

【００１０】一方、データ空間が拡大されても、多数のデータを取り扱うだけで、必ずしも１つの配列の大きさが４バイトの大きさである必要はない。

【００１１】したがって、本発明の目的は、アーキテクチャのデータ幅以上に大きなデータ空間を持つハードウェアをターゲットとする言語処理において、配列要素のアドレス計算を効率良く行い、オブジェクトをより小さくするとともに、オブジェクトの実行性能を向上することにある。

【００１２】

【課題を解決するための手段】本発明の言語処理方法は、ソースプログラム情報を入力するステップと、前記ソースプログラム情報解析するステップと、オブジェクト情報の生成時に処理する配列のアドレス計算をするステップと、前記オブジェクト情報を出力するステップとを有する言語処理プログラムにおいて、前記配列のアド

3

レス計算をするステップが配列の最大サイズを指定したコマンド入力を解析して得たコマンド情報から最短のデータ幅を選択するステップと、前記選択したデータ幅で配列の先頭からのオフセットを求める計算を行なうステップとを有している。

【００１３】

【実施例】次に、図面を参照して本発明の第１の実施例の言語処理方法を詳細に説明する。図１は本実施例のシステムブロック図である。

【００１４】言語処理ステップ１０３は、入力ソースファイル１０１を入力して解析を行なう構文解析ステップ１０４と、コマンド入力１０２を入力して解析するコマンド解析ステップ１０５と、構文解析された内容に従ってコード生成を行なうコード生成ステップ１０６と、生成されたコードをオブジェクトファイル１０９に出力するオブジェクト出力ステップ１０８とから構成される。さらに、コード生成ステップ１０６の中にはコマンド解析ステップ１０５の解析結果に従って適切な計算方法を選択する配列要素アドレス計算ステップ１０７を含む。

【００１５】本実施例では、コマンド入力１０２で入力ソースファイル中に現われる配列の最大の大きさがターゲットマシンのｉｎｔ幅を超えるか否かを指定する。ここではｉｎｔを２バイトとして、配列の最大の大きさが６４Ｋバイトを超えるか否かを指定することになる。コマンド入力１０２の例を図２に示す。“＿ｓｈｏｒｔ＿ａｒｒａｙ”がこの例のコマンド指定である。コマンド解析部１０５ではこの指定を解析して、計算方法指定データとして保存する。

【００１６】構文解析ステップ１０４で配列要素の参照を認識した場合は、まず配列要素のアドレスを計算しなければならないので、コード生成ステップ１０６の配列要素アドレス計算ステップ１０７を呼び出す。配列要素アドレス計算ステップ１０７ではコマンド解析ステップ１０５の解析結果で得られた計算方法指定データに従って、２通りの配列要素アドレス計算アルゴリズムのいずれかを選択する。

【００１７】配列要素アドレス計算１０７の手順を図３を参照して以下に説明する。

【００１８】まず、計算方法指定データが配列がｉｎｔの大きさに収まることを示しているかを調べる（ステップ３０１）。ｉｎｔの大きさに収まるならば生成するオブジェクトとして配列要素アドレス計算アルゴリズム３０４を選択してオブジェクト生成を行なう（ステップ３０２）。ｉｎｔの大きさに収まらないならば従来と同一の配列要素アドレス計算アルゴリズム３０５を選択してオブジェクト生成を行なう（ステップ３０３）。配列要素アドレス計算アルゴリズム３０４は、配列の添字を取り出す（ステップ３０６）。

【００１９】次に、取り出した添字の内容（２バイト）と配列要素のデータ幅の乗算を行なってオフセットを先

4

に求める（ステップ３０７）。ここで配列要素のデータ幅が２バイトを超えるならば複数回に分けて乗算を行なう必要があるが、添字側は２バイトなので４バイト×４バイトの乗算よりも負荷は少ない。オフセットを求めたならばそれをアドレス幅（４バイト）に拡張する（ステップ３０８）。これは次にアドレスとの加算を行なうのでデータ幅を合わせる必要があるからである。以上で配列の先頭からのオフセットが求められる。

【００２０】次に、配列の先頭アドレスを取り出し（ステップ３０９）、取り出した配列の先頭アドレスとステップ３０８で求めたオフセットを加算して配列要素のアドレスを求める（ステップ３１０）。配列計算アルゴリズム３０５は、従来技術と同一の配列要素アドレス計算アルゴリズムである。

【００２１】配列要素参照の例を図４に示す。

【００２２】配列ｔａｂは外部参照をしているので大きさが不明である。そして、“ｔａｂ［ｉ］”についてアドレス計算が必要になっている。これを前述の手順に基づいてオブジェクトを生成した結果を図５および図６に示す。図５は配列の大きさがｉｎｔ以下であることをコマンドで指定した場合、図６は配列の大きさがｉｎｔを超えることを指定した場合を示す。

【００２３】本実施例では、配列要素のデータ幅が４バイトであるので、乗算部分で乗算命令を使用せずにシフト命令を用いている。図５および図６に示す通り、配列の大きさがｉｎｔ以下であることを指定した場合はオブジェクトサイズが合計１９バイト、配列の大きさがｉｎｔを超える場合は合計２０バイトとなる。図６に示された生成コードは従来出力していたものと同じであるから、本実施例において配列の大きさがｉｎｔ以下であることを指定した場合は。従来のオブジェクトサイズよりも１バイト減少することがわかる。

【００２４】次に、本発明の第２の実施例の言語処理方法について説明する。

【００２５】上記第１の実施例は、コマンド入力によって配列の大きさがｉｎｔを超えるか否かを指定させたが、全ての配列がｉｎｔ以下／ｉｎｔより大きいのではなく配列毎にｉｎｔを超えるか否かが異なる場合がある。この場合は、入力ソースファイル中にコマンドを埋め込む手段によって最適な配列要素アドレス計算アルゴリズムを選択することが可能である。

【００２６】図７を参照すると言語処理ステップ７０２は、構文解析ステップ７０３の中にコマンド解析ステップ７０４が含まれること、入力ソースファイル７０１以外の入力データが存在しないこと以外は、第１の実施例のステップと同じステップを有している。

【００２７】本実施例では図８に示す通り、配列の大きさがｉｎｔを超えるか否かを入力ソースファイル７０１に中に記述する。図８に示す配列ｔａｂ１は配列の大きさがｉｎｔ以下であることを示すコマンド”ｓｈｏｒｔ

5

－ａｒｒａｙ記号が宣言部に記述されている。

【００２８】構文解析ステップ０３が配列の宣言を認識したならば、コマンド解析ステップ７０４でｉｎｔを超えるか否かのコマンド指定の有無から、その配列の大きさがｉｎｔを超えるか否かを判断する。そして、コマンド解析ステップ７０４の判断に基づき、配列要素の参照を認識した場合は、コード生成ステップ７０５の配列要素アドレス計算ステップ７０６を呼び出す。以降の処理については第１の実施例と同じ処理であるので詳細な説明は省略する。

【００２９】この第２の実施例は、配列のそれぞれに対して配列の大きさを指定できるので、第１の実施例よりもきめ細かなアドレス計算方法の選択が可能である。

【００３０】

【発明の効果】以上、説明したように本発明によれば、コマンドによって指定された配列の大きさに対応して最適な配列要素のアドレス計算アルゴリズムを選択することにより、生成するオブジェクトのサイズを短縮することが可能である。

【００３１】配列の大きさがｉｎｔ以下である場合を例にとると、第１の実施例で説明した通り、コマンド指定により１回の配列要素のアドレス計算につき１バイト短縮することが可能である。第１の実施例のデータはＣ言語のｌｏｎｇデータ（４バイト）のシフト演算が存在するマシンアーキテクチャをターゲットとしたものだが、ｌｏｎｇデータのシフト演算が存在しないマシンの場合は従来技術の計算方法ではｌｏｎｇデータのシフトをライブラリコールによって実現することになり、増加するオブジェクトサイズ、増加した命令の実行時間分、本発明による計算方法の法が有利となる。

【００３２】すなわち、生成するオブジェクトのサイズを短縮することにより、プログラム空間を節約することができるとともに、オブジェクトのロード時間の短縮を

6

図ることも可能となる。

【図面の簡単な説明】

【図１】本発明の第１の実施例の言語処理方法全体の構成図である。

【図２】図１に示す言語処理方法のコマンド入力の例を示す図である。

【図３】図１に示す言語処理方法の配列要素アドレス計算アルゴリズム選択処理の流れ図である。

【図４】入力ソースファイルの例（第１の実施例）を示す図である。

【図５】配列要素アドレス計算アルゴリズム３０４による出力オブジェクト例を示す図である。

【図６】配列要素アドレス計算アルゴリズム３０５および従来技術による出力オブジェクト例を示す図である。

【図７】本発明の第２の実施例の言語処理方法全体の構成図である。

【図８】図７に示す言語処理方法の入力ソースファイル例を示す図である。

【図９】従来の言語処理方法の配列要素アドレス計算アルゴリズムの流れ図である。

【符号の説明】

１０１，７０１　入力ソースファイル
１０２　コマンド入力ステップ
１０３，７０２　言語処理ステップ
１０４，７０３　構文解析ステップ
１０５，７０４　コマンド解析ステップ
１０６，７０５　コード生成ステップ
１０７，７０６　配列要素アドレス計算ステップ
１０８，７０７　オブジェクト出力部ステップ
１０９，７０８　オブジェクトファイル
３０１～３１５　配列要素アドレス計算アルゴリズム選択処理
９０１～９０５　従来の配列要素アドレス計算処理

【図２】

```
ccxx  -short_array …  入力ソースファイル名
```

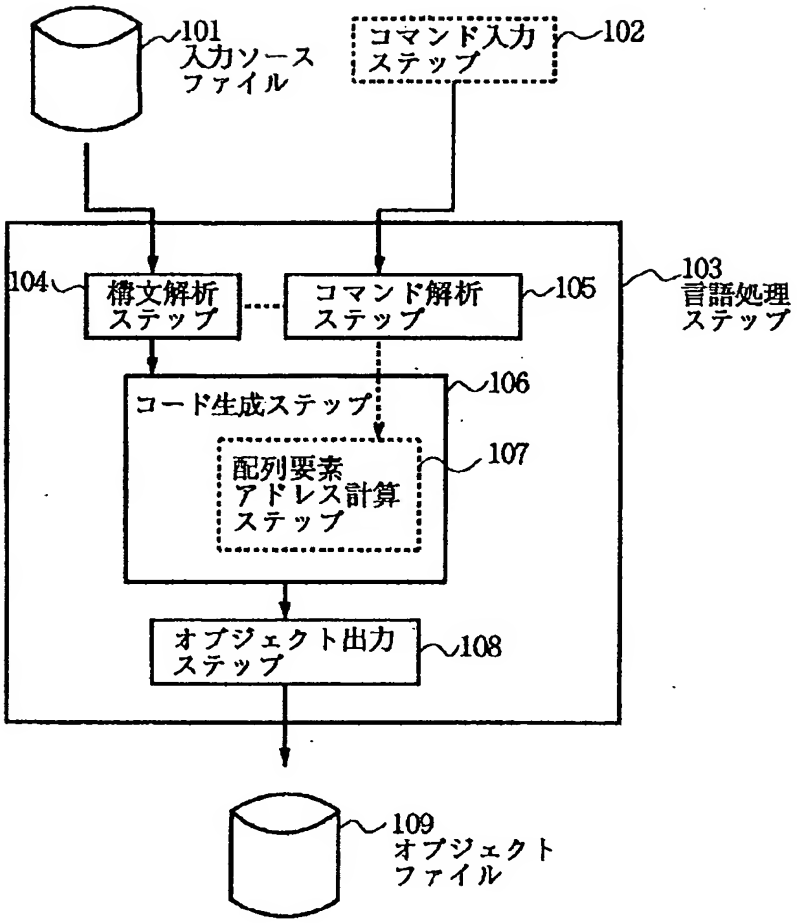
【図４】

```
extern long tab[];
void func(int i)
{
        tab[i]= 1;
}
```

【図５】

```
movw   r0,B:8[rp5]     ;i     (4バイト)
shlw   r0,04                  (2バイト)
cnvwd  rp0                    (2バイト)
addd   rp0,#_tab              (6バイト)
movw   [rp0],#1        ;1     (5バイト)
```

【図１】



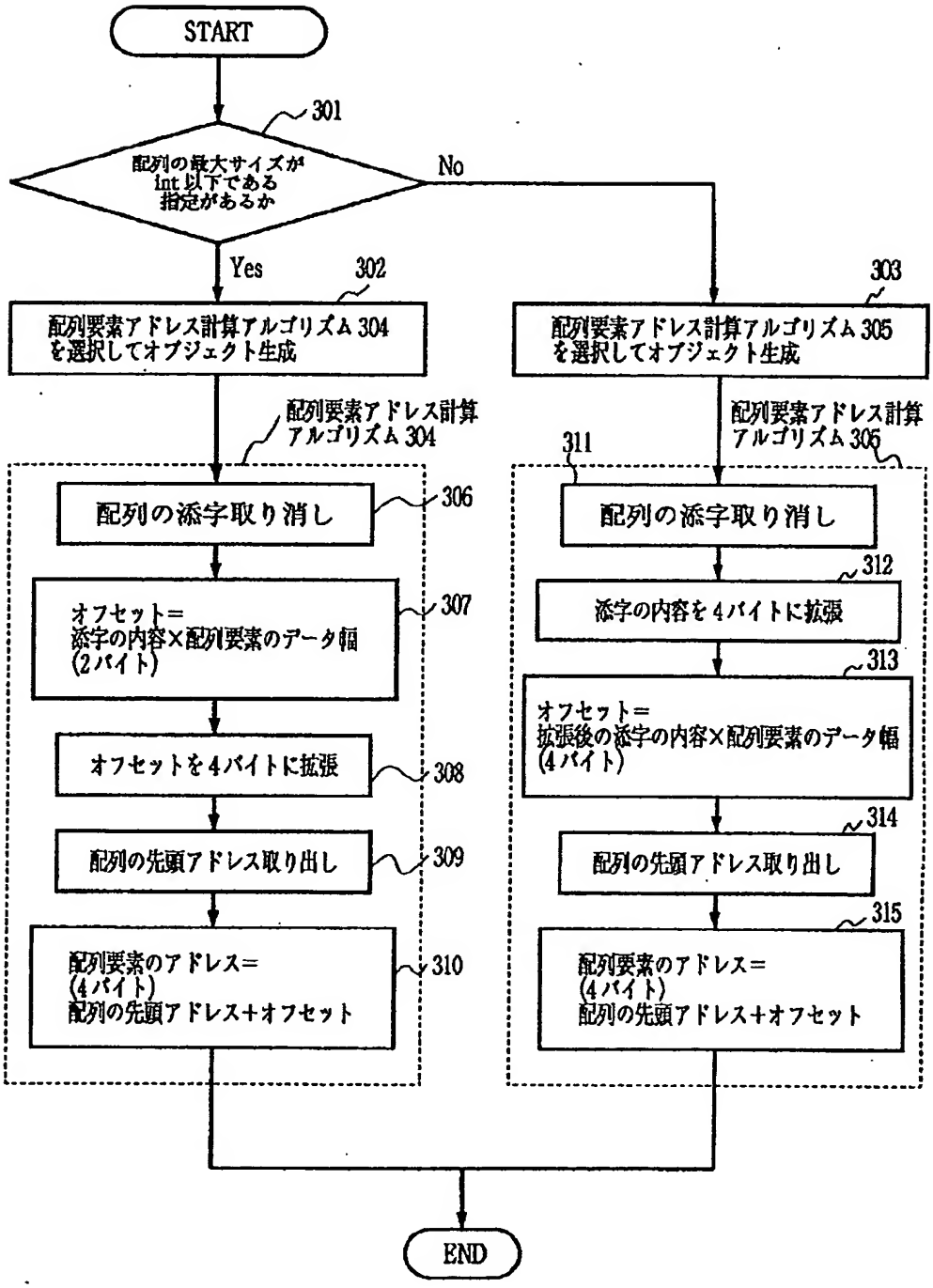

【図６】

```
movw   r0,B:8[rp5]    ;i      (4バイト)
cnvwd  rp0                     (2バイト)
shld   rp0,04                  (3バイト)
addd   rp0,#__tab              (6バイト)
movw   [rp0],#1       ;i       (5バイト)
```

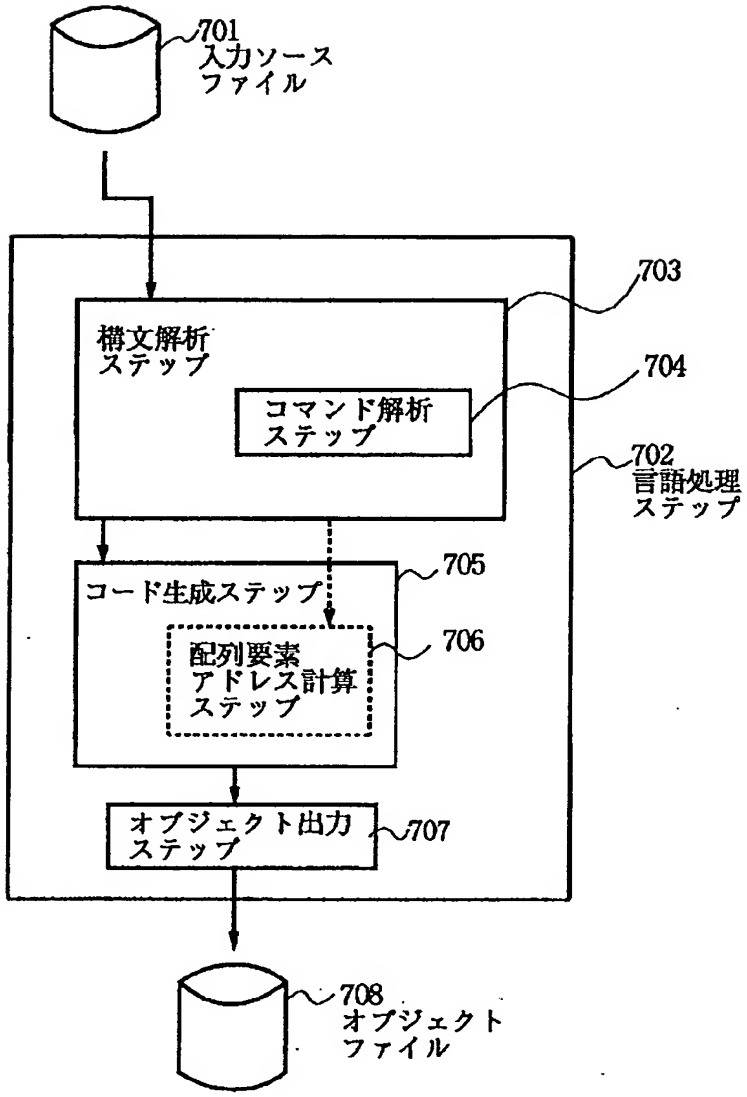
【図８】

```
short__array extern long tab1[];
extern int tab2[];
void func  (int i, int j)
{
        tab1[i]= 1;
        tab[j]= 1;
}
```

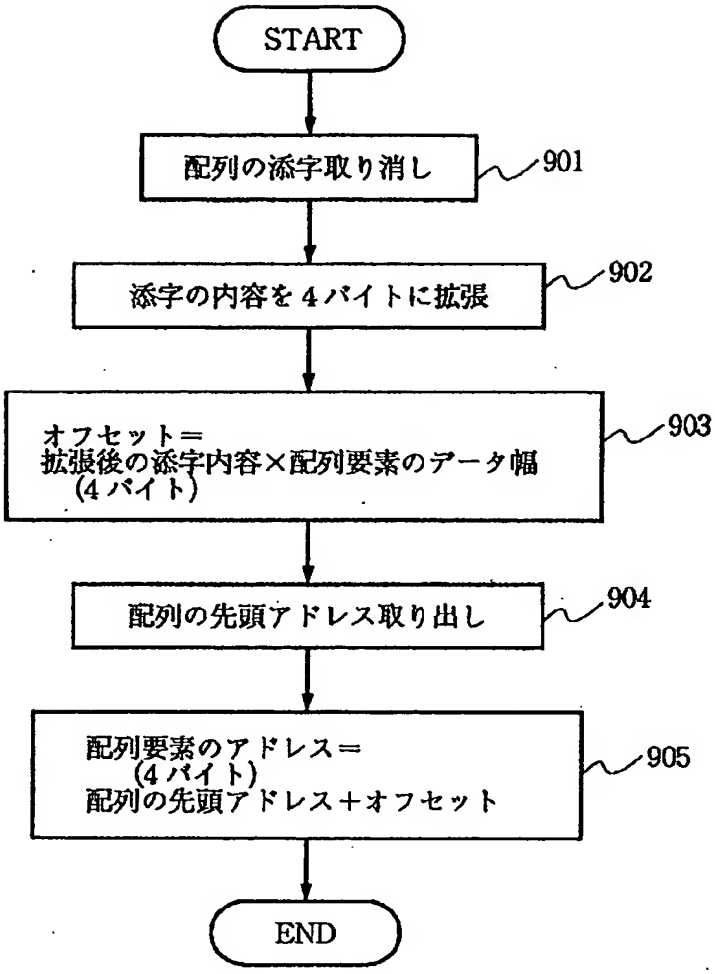
【図３】


START
301
配列の最大サイズが
int以下である
指定があるか
No
Yes
302
303
配列要素アドレス計算アルゴリズム304
を選択してオブジェクト生成
配列要素アドレス計算アルゴリズム305
を選択してオブジェクト生成
配列要素アドレス計算
アルゴリズム304
配列要素アドレス計算
アルゴリズム305
311
配列の添字取り消し
306
配列の添字取り消し
312
添字の内容を4バイトに拡張
オフセット=
添字の内容×配列要素のデータ幅
(2バイト)
307
313
オフセット=
拡張後の添字の内容×配列要素のデータ幅
(4バイト)
オフセットを4バイトに拡張
308
314
配列の先頭アドレス取り出し
309
配列の先頭アドレス取り出し
315
配列要素のアドレス=
(4バイト)
配列の先頭アドレス+オフセット
310
配列要素のアドレス=
(4バイト)
配列の先頭アドレス+オフセット
END

【図７】


701
入力ソース
ファイル
703
構文解析
ステップ
704
コマンド解析
ステップ
702
言語処理
ステップ
705
コード生成ステップ
706
配列要素
アドレス計算
ステップ
707
オブジェクト出力
ステップ
708
オブジェクト
ファイル

【図9】


START
配列の添字取り消し
901
添字の内容を4バイトに拡張
902
オフセット＝
拡張後の添字内容×配列要素のデータ幅
(4バイト)
903
配列の先頭アドレス取り出し
904
配列要素のアドレス＝
(4バイト)
配列の先頭アドレス＋オフセット
905
END